

Consistency Maintenance of State of Management Data in P2P-based Autonomic Network Management

Jéferson Campos Nobre*, Lisandro Zambenedetti Granville

*Institute of Informatics - Federal University of Rio Grande do Sul
Av. Bento Gonçalves, 9500 - Porto Alegre, RS - Brazil*

Abstract

Complex Dynamic Networks can be exploited in solving problems where traditional solutions may not be sufficient. The increasing complexity of computer networks imposes problems to the current network management solutions. In this context, network management is an example of a research area that could benefit from the use of CDNs. However, the consistency of state of management data among the elements that build management CDNs (management nodes) is an important challenge. Traditional mechanisms to maintain consistency of these states are supported by some centralization which wastes some desirable properties of CDNs (*e.g.*, robustness). In contrast to these mechanisms, we propose a distributed, scalable and robust mechanism to maintain the consistency of state of management data in management CDNs. Our mechanism introduces multi-agent truth maintenance features and communication strategies based on dynamic process to provide consistency maintenance of state of management data. We developed a model of a management CDN on Peersim simulator to perform experiments. Besides, 2 case studies are presented. The result obtained supports our scalability and robustness claims.

Keywords: Network/service operations and management, DTMS, MAS

1. Introduction

Complex Dynamic Networks (CDNs) are networks whose dynamic and collective behavior is a function of the sophisticated individual properties of nodes and links that compose them [42]. CDNs are found and observed in several fields, varying, for example, from biology (*e.g.*, immune systems) to computer science (*e.g.*, social networks). As such, different research initiatives have been investigating how to model and analyze CDNs in order to better understand their behavior [31]. Inspired by studies of real-world CDNs, such initiatives have identified important CDN characteristics, such as network evolution and dynamical complexity [42]. From another perspective, CDNs can also be exploited

*Corresponding author.

Email addresses: jcnobre@inf.ufrgs.br (Jéferson Campos Nobre), granville@inf.ufrgs.br (Lisandro Zambenedetti Granville)

Preprint submitted to Computer Networks

in solving problems where traditional solutions may not be sufficient. The management of computer networks is an example of a research area, which is key to this article, that could benefit from the use of CDNs.

The complexity of computer networks, as well as their use and importance, has increased significantly in recent years. However, computer network advancements are usually not accompanied by corresponding adequate management solutions [39]. One possibility of tackling this problem is by incorporating CDN characteristics into network management, especially if one considers that these characteristics (*e.g.*, improved fault tolerance, distributed processing, and load balance) are usually expected, but not always found, in today's network management systems. In fact, some management solutions proposed by the network management community (*e.g.*, mobile agents and peer-to-peer for network management) can be described as CDNs themselves. The recent use of CDNs concepts in network management, however, also introduces new problems that need to be addressed.

A CDN for network management is an overlay network running on top of the communication infrastructures that need to be managed. A management CDN is composed of peers that have a double role: besides acting as regular CDN nodes (*i.e.*, performing operations to support the CDN), they also execute management tasks over the underlying managed infrastructure. CDN nodes themselves need to store and exchange data as a part of these tasks. However, the state of management data stored in these nodes can become inconsistent because of, for example, the dynamicity of nodes' interaction. For example, in automated and distributed decision making processes, management decisions must be advertised by different nodes of the management CDN in a coordinated and coherent way. These advertisements are directed towards the interested human administrators or management applications. If incoherent advertisements take place (*i.e.*, some nodes advertise different results for the same decision), they can lead to instabilities in the underlying network management. Thus, a mechanism to maintain the consistency of state of management data is necessary.

Despite the importance of having consistency mechanisms for the state maintenance of management data in today's management CDNs, this topic has been barely addressed in previous investigations, as will be further discussed in Section 2. In fact, when present, such mechanisms are in general materialized in centralized solutions [3] [9], which hinder distribution benefits (*e.g.*, scalability), in addition of being essentially inconsistent with the distributed nature of CDNs. Even when distribution indeed takes place, these mechanisms are usually not clearly described in the distributed management literature [17] [25].

In this article we present a solution for the maintenance of state consistency of management data in management CDNs. The contribution of this article is twofold. First, we propose the use of *belief exchange* about management data to improve the consistency of states of management data in a management CDN. The proposed belief exchange is inspired by multi-agent truth maintenance [15], a concept that we borrow from Multi-Agent Systems (MAS) [43], where states of management data are organized as a set of justified beliefs among the management nodes. Second, we present communication strategies in order to support belief exchange among management nodes. These strategies use dynamic bio-inspired processes (proliferation), which are recognized as scalable and robust [2]. The employment of these communication strategies creates multi-layered management CDNs.

The remainder of this article is organized as follows. In Section 2 we present a motivating scenario, while in Section 3 we detail how the concept of belief exchange is adapted to consistency maintenance of management data in the context of management CDNs. We describe the architecture of a management node in Section 4 and we show case studies in Section 5. We introduce the communication strategies for the belief exchange among management peers in Section 6. In Section 7 we evaluate our proposal discussing the results obtained with simulation experiments. We compare our research with related work in Section 8 and finally close this article with concluding remarks and future work in Section 9.

2. State consistency for management CDNs: P2P-based autonomic network management as a motivating scenario

There has been substantial research on autonomic features related to network management solutions [39]. The application of Autonomic Computing (AC) principles in network management, normally referred as *Autonomic Network Management* (ANM), has been proposed as a way to address some demands faced by traditional network management, such as controlling highly dynamic environments as in *ad hoc* networks [37]. Several authors claim that some level of decentralization plays an important role to perform autonomic actions in a more adequate manner (*e.g.*, [6]). Different technologies could be employed as an infrastructure of a decentralized ANM system. An interesting possibility is using P2P overlays, which incorporates characteristics of P2P-based network management (P2PBNM) into ANM systems, such as the robustness in connectivity of management entities [13].

The constitutive peers of a P2P-based ANM system interact dynamically and behave collectively, given their individual dynamics and coupling pattern. Thus, they can be defined as management nodes of a management CDN. These nodes expose dynamic properties because of their autonomy capabilities in the execution of management tasks. Other examples of systems taking the form of CDNs abound in the world, such as, social networks, networks of business relations between companies, metabolic networks, food webs, blood vessels, power grids, and networks of citations between papers [42] [31].

The execution of management tasks can change the state of management data among nodes. However, this state must be consistent across a management overlay. Thus, consistency maintenance procedures play an important role in management CDNs. These procedures must handle heterogeneity in these CDNs. Management nodes can be heterogeneous, being deployed in management stations or embedded into network devices. Besides, management tasks can be also heterogeneous, being requested by human network administrators or triggered by state-of-the-art Machine-to-Machine (M2M) interactions [22]. A biological analogy that can be used is the coherent and stable behavior from cells that form tissues of complex multicellular organisms.

In this Section, we present an overview of a typical consistency maintenance procedure in a P2P-based ANM system. Afterwards, the most significant P2P-based ANM initiatives found in the literature and their consistency maintenance mechanisms are described.

2.1. A typical consistency maintenance procedure in a P2P-based ANM system

Before introducing the issues associated to a typical consistency maintenance procedure, consider the environment of a decentralized ANM system. It is often composed of a set of Autonomic Management Elements (AME), and these AMEs are often grouped into Autonomic Management Domains (AMD). In P2P-based ANM initiatives, peers have some properties found in AMEs and peer groups have some properties found in Autonomic Management Domains (AMD). The interactions among AMEs of an AMD can be characterized as a management CDN.

P2P-based ANM systems use the P2P overlay communication services to propagate messages among the peers. This overlay can be modeled as an asynchronous unreliable communication network, thus it must be considered some issues, such as message delay and loss. In this context, P2P-based ANM systems can be described as asynchronous distributed systems. Besides, it is well known that the utilization of asynchronous distributed systems imposes challenges to achieving consensus or agreements [10]. Thus, the consistency maintenance mechanism aims in overcome these challenges to provide support for the consistency maintenance of states of management data. To illustrate a consistency maintenance procedure, it is described a change in the state of a specific management datum due the reception of an asynchronous signal by a decentralized ANM system. In this example, this change - a new state of a management datum - must be consistently stored in the system's Knowledge Base (KB).

The traditional consistency maintenance mechanism is the utilization of a shared repository for storing states of management data. Thus, state of management data is centrally-stored, whether in a special peer or even in a regular server (*i.e.*, any single point of storage). This mechanism can be seen in some P2P applications through the use of a "central server" (*e.g.*, Napster). The consistency maintenance procedure performed by this mechanism is defined as follows. After receiving an asynchronous signal, the peers send a message to the server to inform the change in the state of management data. After these messages, the server sends a message to peer group and peers check whether the state is coherent with their KB. If it is not, the peers update their KB.

There are some major issues related to such centralized approach to deploy this consistency maintenance mechanism. First, the utilization of a shared resource presents robustness issues, since the shared repository is a single point of failure (SPoF). As a consequence, this repository is often implemented using clusters. Second, there are robustness issues because the shared repository represents a bottleneck in terms of computation and bandwidth. This may translate into increases in the total cost of ownership (TCO) to support a bigger management infrastructure. Third, this centralization decreases peers' autonomy. This can derail the operation of a fully decentralized ANM system. These issues underscore the utilization of shared repositories to maintain the consistency of states of management data.

2.2. P2P-based ANM initiatives and their consistency maintenance mechanisms

There are several initiatives investigating P2P-based ANM. These initiatives have different consistency maintenance mechanisms. For sake of simplicity, only few initiatives in P2P-based ANM will be cited, as follows.

PBMAN [20] merges traditional PBNM with P2P overlays to manage Ambient Networks (AN). PBMAN enables management tasks inside the AN, as well as distribution

and retrieval of management policies. Through this approach it is possible to manage seamlessly devices or services. PBMAN is structured using super peers, in a hierarchical architecture. States of management data are not effectively distributed in this initiative, and, for fault tolerant features, super peers are replicated.

ManP2P [35] is an example of P2P-based network management system that is evolving to an autonomic conception through the implementation of autonomic modules in peers. ManP2P is partially inspired by the Management by Delegation (MbD) model and based on a service-oriented approach. The autonomic modules are designed to support self-* features and to communicate with other components of the ANM system, when necessary. The distribution of states of management data is not clearly described in this initiative [25].

Self-Managed Cells (SMC) [24] are proposed as an architectural pattern for ubiquitous computing applications, aiming at different levels of scale. Each SMC is autonomous and uses policy-based techniques for driving adaptation decisions. Among different cross-SMC interactions, the authors describe P2P interactions. But, concerning to levels of abstraction, it is not clear whether different SMCs could be “peers”. A managed device is logically connected with only one SMC, thus, the state of management data is not evaluated in a P2P fashion.

The Madeira platform [9] is an approach to network management topologies that uses the concept of Adaptive Management Components (AMC), which are containers that run on managed elements. AMCs can manage elements on which they are running and communicate with other AMCs running on other managed elements through P2P communication services. AMCs form management clusters, where a cluster head (*i.e.*, super peer) coordinates the cluster and correlates data. The use of cluster heads centralizes the maintenance of state of management data.

Despite many improvements brought by P2P-based ANM systems (*e.g.*, higher availability of network management system), there are still issues to be addressed. States of management data in different peers of an P2P-based ANM system can become inconsistent in overlay operation due to faults or lack of synchronization. Besides, the consistency maintenance of these states must be done maintaining scalability and robustness features of P2P overlays. This consistency maintenance is equally important to any management CDN to present a coherent behavior as a whole.

The scenarios faced by management CDNs (exemplified by P2P-based ANM systems in this article) to maintain consistency of states of management data are similar to those faced in different Multi-Agent Systems (MAS) [43]. The operation of a weakly coupled infrastructure hampers the utilization of conventional strategies such as explicit message exchange or shared memory for coordination purposes [5]. In a MAS, agents must be able to assess and maintain the integrity of the information exchanged among them and conflicts about contradictory knowledge may arise during the communication. Besides, agents have to attend their tasks in an asynchronous way. Thus, techniques from MAS could be used to improve the consistency of states of management data in management CDNs.

3. Justifications for Management Data

In a management Complex Dynamic Network (CDN), nodes that perform a specific task must share management data. In this article, *management datum* is defined as a management information described in a defined form (*i.e.*, using a specific language). Management CDNs use several sets of management data in their operation, which can be integrated into *Knowledge Bases* (KBs). The consistency maintenance of these KBs is one of the main challenges related to management CDNs, such as P2P-based ANM systems. In these CDNs, KBs are distributed (and shared) among the elements that build these networks (*i.e.*, management nodes).

Management data must allow their use in distributed automation and/or optimization procedures in a management CDN, which hampers the consistency maintenance of these data. For example, learning and reasoning techniques in distributed management ontologies can be present in these procedures [17]. It is also expected that sources of management data (*e.g.*, highly dynamic environments) impose challenges to the management CDN. Despite these challenges, it is necessary to avoid potential inconsistencies in the state of management data among nodes. This state among different nodes builds another layer over the management CDN, *i.e.*, creating a multi-layer overlay.

Our proposal is aimed at meeting consistency requirements of state of management data in a management CDN. The proposed mechanism introduces *multi-agent truth maintenance* [15] features through a *consistency maintenance module* (this module is described in Section 4) that runs in each node. These features are represented by belief exchange about management data through the utilization of justifications. In this context, management nodes begin to display some characteristics of intelligent agents and the management CDN aggregates some features related to Multi-Agent Systems (MASs). As far as we are aware of, the only studies that incorporate multi-agent truth maintenance features in network management were carried out by Nobre and Granville [33] [32] [34].

In this Section, we present the utilization of *justifications for management data*. Initially, we present an overview of truth maintenance in order to provide a theoretical background (Subsection 3.1). Afterwards, our conceptual solution and an internal representation are described (Subsection 3.2).

3.1. Truth Maintenance

Truth Maintenance Systems (TMSs) were proposed to keep the integrity of KBs. The origin of these systems was proposed in the 1970s, for resolutions in mono-agent systems [7] [8]. TMSs provide considerable power using few computational resources [18]. Although not being well known outside artificial intelligence community, TMSs are used in different contexts, such as policy systems [18], recommendation systems [23], plan adaptation and repair [45], and ontology schemes [4].

TMSs keep the integrity of KBs performing belief revision and exchange in a set of beliefs. These systems keeps track of logical structure of the set of beliefs of agents. A belief is a member of the current set of beliefs if it has valid reasons (*i.e.*, the belief is well-founded). Usually, TMS is implemented through a software module inside an agent. The evaluation of logical consistency of the KB is done by another module, the Problem Solver (PS) [15]. The PS sends beliefs and their respective foundations to TMS, which,

then, registers and associates foundations with respective beliefs. For instance, in policy systems, the PS role is played by the software component that handles policy processing.

TMSs have been extended for MAS versions (*e.g.*, Distributed Truth-Maintenance Systems [15]). These multi-agent extensions are commonly referred as *multi-agent truth maintenance*. In a MAS, agents must be able to maintain the integrity of their KBs, despite message exchange with other agents. In order to perform truth maintenance, each agent has its own TMS, thus maintaining the integrity of their KBs.

Each agent keeps its local data structure. However, agents can be heterogeneous, since agents can have distinct data sets. In this context, data inside agents can be divided in 2 classes: shared data, beliefs that the agent has shared with another agent at some time in the past; and private data, beliefs that the agent has never shared with another agent [15]. The relation between shared and private data among agents is shown in Figure 1.

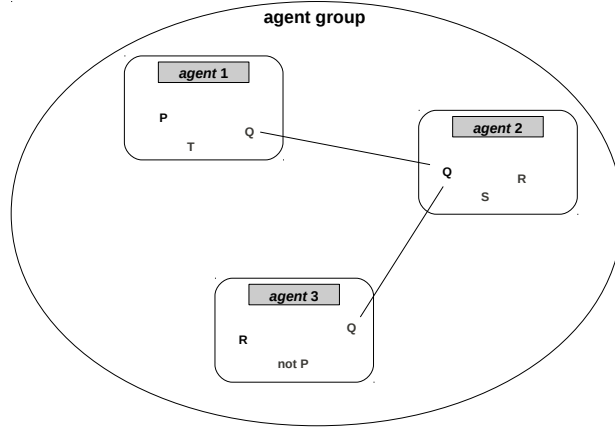


Figure 1: Shared and Private Data - Figure adapted from [15]

Figure 1 describes a group of 3 agents in a given moment. Besides, it is shown some data stored by each node. The “Q” datum is the only one shared inside this agent group. The other data represented in this Figure (“R” and “T”) are private with respect to specific agents. Agents need to be consistent just about data they all share. Thus, according to the agent group described in Figure 1, only beliefs about the “Q” datum would be exchanged inside this group.

One of the possible alternatives to implement multi-agent truth maintenance features is the utilization of *justification-based* TMSs [15]. In a justification-based TMS, a datum is believed when it has valid justifications (*i.e.*, valid reasons). Justifications are produced according to available information, in a defined way (*i.e.*, understood by the TMS) [8]. These justifications can be shared among different agents in a MAS.

3.2. Conceptual Solution

Nodes must be able to maintain the integrity of states of management data during the operation of a management CDN, despite message exchange with other nodes. This similarity indicates the use of multi-agent TMS in management CDNs as an interesting possibility. In this Subsection, a conceptual solution is proposed to allow that beliefs can be shared by different nodes (which play the agent role) in a management CDN (which aggregates some MAS characteristics) through *justifications*. In contrast to previous initiatives, our solution focuses on providing a distributed way to maintain the consistency of state of management data by using multi-agent truth maintenance. Justifications improve the alignment of nodes with system-wide objectives (*i.e.*, objectives of the management CDN). Besides, the structure of beliefs and their justifications is a CDN itself.

The datum and its list of possible justifications must be provided by network human operators or expert systems for the management CDN. The associated states of a datum are “in” (believed) or “out” (disbelieved), according to the presence of its justifications. A datum is labeled “out” when it lacks at least one of its associated justifications. We assume that there are not contradictions, so, a datum is labeled “out” only if it has not any of its associated justifications.

In the seminal work from Doyle [8], a TMS does not require a list of possible justifications for a datum, using an *ad hoc* approach. In order to simplify the implementation and analysis issues, we defined that the datum and its list of necessary justifications must be provided *a priori*. The datum and its list of possible justifications must be provided by network human operators or expert systems. The presence of justifications are based in management information processed by nodes of the management CDN.

For instance, the activation (belief) of a QoS policy (datum) can be justified by a network human administrator command (justification) and an asynchronous signal from a managed device (justification). Listing 1 shows a possible internal representation of this datum and its justifications using ISO *Prolog* standard [41]. In the example, justifications and their presences are represented by facts (Prolog notation) and the datum is represented by a rule (Prolog notation). The last line of Listing 1 defines that if the justifications “adm_cmd” and “async_sig” are present, the datum “qos_pol” is believed.

```

1      justification(adm_cmd).
2      justification(async_sig).
3      justificationIsPresent(X) :- generated(X).
4      justificationIsPresent(X) :- received(X).
5      datumIsInternal(qos_pol) :-
6          generated(adm_cmd),
7          generated(async_sig).
8      datum(qos_pol) :-
9          justificationIsPresent(adm_cmd),
10         justificationIsPresent(async_sig).
```

Listing 1: Activation of a QoS policy

Justifications can be generated by processes inside the node or received through the CDN communication services. Thus, the “in” state can assume two additional states: “internal”, where the datum has only valid internal justifications, and “external”, where the datum has some valid external justification (provided by other node) [15]. These

additional states are enabled by “justificationIsPresent” rules as shown in Listing 1. Thus, the presence of a justification is inferred by “generated” or “received” facts.

The current state of a datum can be checked during the operation of the management CDN. This state is produced through the presence of justifications and their sources. Listing 2 shows a possible response using the example described in Listing 1 (activation of a QoS policy). This response denotes that the “qos_pol” datum is believed (*i.e.*, “in”). The absence of any of the justifications would change the state of the datum to disbelieved (*i.e.*, “out”).

```
1      qos_pol : internal (adm_cmd : mod async_sig : mod)
```

Listing 2: A response from consistency maintenance module

The response shown in Listing 2 also indicates the option “internal” being checked. This response shows that the presence of all the justifications was internally generated (indicated by “:mod” in Listing 2). If the presence of some justification was inferred through belief exchange from other nodes (which would be indicated by “:msg”), the option “external” would be checked.

Justifications can be also used in an hierarchical way, thus a datum can be used as a justification for other data. Thus, the presence of this kind of justification is controlled by the state of the datum, in other words, if the datum is “in” it appears as *present* and if the datum is “out” it appears as *absent*. This feature provides support for the representation of more complex data.

The example shown in Listing 1, the activation of a QoS policy, can be modified to show the utilization of hierarchical justifications. We extend the justification “adm_cmd” (“Administrator Command”) in order to embrace 2 different commands from a human administrator. Thus, we define a new datum “adm_cmd” and these 2 different commands are represented by 2 new justifications “adm_cmd1” and “adm_cmd2”, according to Listing 3

```
1      justification (adm_cmd1) .
2      justification (adm_cmd2) .
3      justificationIsPresent (X) :- generated (X) .
4      justificationIsPresent (X) :- received (X) .
5      datum (adm_cmd) :-
6          justificationIsPresent (adm_cmd1) ,
7          justificationIsPresent (adm_cmd2) .
8      datumIsInternal (adm_cmd) :-
9          generated (adm_cmd1) ,
10         generated (adm_cmd2) .
```

Listing 3: A new “Administrator Command” version

Modifications are also necessary in “QoS Police” datum (qos_pol). Listing 4 shows these modifications. The new “Administrator Command” datum (adm_cmd) is used as a justification for the “QoS Police” datum through a modification in the “datum(qos_pol)” rule. Besides, the “datumIsInternal(qos_pol)” rule is also modified to allow the option “internal” to be properly marked. Thus, if every justification of a datum is internally generated, the datum is processed also as an internally generated justification.

```
1      datum (qos_pol) :-
2          datum (adm_cmd) ,
```

```

3      justificationIsPresent( async_sig ).
4      datumIsInternal( qos_pol ) :-
5          datumIsInternal( adm_cmd ),
6          generated( async_sig ).

```

Listing 4: A new “QoS Police” version

The conceptual solution for justifications described in this article does not include the representation of justifications that identify belief absences. Traditional TMSs operate with 2 lists: the “IN” list, which depicts the accredited beliefs, and the “OUT” list, where are represented the discredited beliefs [8]. Thus, the representation of an absence is defined as part of the “OUT” list. In this proposal, there is only a list of justifications, which it is related to accredited beliefs (“IN” list). However, the definition of a justification that identifies an absence can be performed by the human administrator, using, indirectly, the meaning of the justification. Thus, for example, the absence of a command from an administrator could be described as an “Administrator Command Off” fact.

4. Architecture of management nodes

Management nodes can be viewed as a *software container* for one or more *management service modules*. These modules perform regular management tasks (*e.g.*, collecting statistics) in each management node, and, in these tasks, modules produce management information. Furthermore, these nodes can appear and disappear in normal operation, behavior that conventional network management systems do not expect from their constitutive elements.

We introduce the *consistency maintenance module* to register and handle the set of beliefs about management data in each management node. This module works associating management data and their respective justifications. In order to simplify the implementation and analysis issues, we define that the consistency maintenance module must receive the datum and its list of necessary justifications (structure).

When there is a belief change (*i.e.*, a justification change), the consistency maintenance module uses the CDN communication services to spread the change. Figure 2 shows the relation between the consistency maintenance module, management service modules, and CDN communication services.

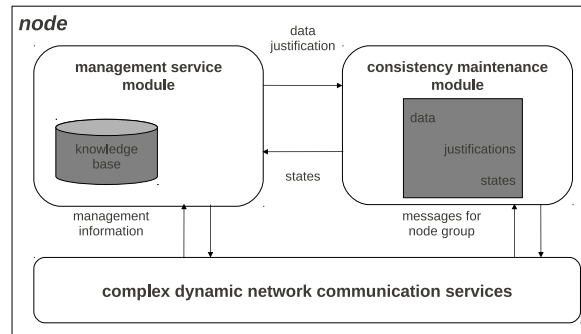


Figure 2: Architecture of a management node

Beliefs can be represented using different languages. *A priori*, the consistency maintenance module does not require a particular language for belief representation. However, the internal representation of management data and justifications must be unique among the management nodes. In Section 3, we present a representation using facts and rules, concepts from ISO Prolog language.

It is important to warn that the responsibility of logical consistency of a datum and its justifications is not in charge of the consistency maintenance module. The consistency maintenance module verifies whether the defined justifications for a certain datum are present, and, using this information, controls the state of this datum. An advantage of this approach is that the consistency maintenance module is only triggered when there is a change in belief status of shared data, which means it does not introduce overhead for regular management tasks.

The management service modules should inform the consistency maintenance module about their internal beliefs in respect to management data. When the presence of a justification is modified (internally generated or received through a belief exchange message) the consistency maintenance module performs the following steps: unlabels the management datum, includes (or removes) the presence of justification and labels the datum again according to new restrictions. The CDN communication services are used to spread changes, which can change beliefs of other management nodes.

The management service modules are responsible for querying and requiring services from the consistency maintenance module, according the demands of their processes. Some similar services employ different strategies such as “*publish/subscribe*” schemes [44] or “*watches* (changes trigger the transmission of a packet) [16]. These strategies could be implemented in the proposed consistency maintenance module, however, initially we keep this operation in charge of management service modules.

It is important to stress that there is only one consistency maintenance module inside a management node, thus, it is not specific to a management service module. Therefore, every management service module in this node interacts with the same consistency maintenance module. This fact can be explored for the integration of different management services. For instance, a policy processing module, a fault handling module, and a configuration management module (possibly using different languages for representing management data) could be integrated by the consistency maintenance module through justifications.

The services offered by management service modules and the consistency maintenance module may be locally available through the use of interprocess communication. These services can be offered in different ways, such as, for example, using a message bus like *D-Bus* (*Desktop Bus*) [36] or through interprocess communication protocol like *DCOP* (*Desktop COmmunication Protocol*) [30].

The consistency maintenance module is conceived considering that there is support for group organization (*i.e.*, node groups) through management services modules. Thus, nodes that have a specific management service module are organized into a group (without human intervention) and nodes can participate of several groups accordingly to modules that they have [25]. In this context, the management CDN must support group communication services. These services are often supported in overlay networks. For instance, the “PeerGroup Service” from JXTA *framework* [12] is an implementation for an partially structured overlay. An interesting possibility to support group communication services is the utilization of *multicast*, but this utilization is not a *sine qua non* condition.

The consistency maintenance module offers an open environment for rapid and dynamic resource integration of management service modules are formed with no central authority. This is feasible because each node runs its own consistency maintenance module, which keeps its own data structure and handles belief exchange through messages among other nodes. These groupings have analogous characteristics with *federations*, a concept brought from MAS literature [14].

The consistency maintenance module is offered through a simple interface that management service modules must use. The number of supported operations is restricted to ease the implementation in management service modules. Besides, communication details are not specified in supported operations, so changes in communication strategies do not lead to adaptations in the interfaces of management service modules.

5. Case studies

Case studies can be used to illustrate the utilization of justification for management data through the consistency maintenance module in a management CDN. Network management, in which we aimed at the present proposal, is an interesting context for CDNs because of their desirable characteristics. For example, one of this characteristics is that some CDNs display a remarkable degree of tolerance against errors [1]. However, the consistency of state of management data can impose challenges to management CDNs (as discussed in Section 2).

In this Section, 2 case studies are presented. In the first subsection, the consistency maintenance of state of obligation policies in the form of Event-Condition-Action rules is described. In the following subsection, collaborative fault management of links in access networks through failure notification sent by devices and human knowledge about these notifications is characterized.

5.1. Consistency maintenance of state of obligation policies

The first case study presented is an illustration of the consistency maintenance of state of policies in a management CDN (a simpler and shorter version of this case study was described in a previous work [34]). We use obligation policies in the form of Event-Condition-Action (ECA) rules as our model. Using the concepts from our proposal, policy state (*i.e.*, policy activation) is a management datum, and event and conditions are justifications in the consistency maintenance module. This module provides an integration point to enable a coherent global behavior of policy-driven (or -enabled) management CDNs.

We describe the policy used in this case study using *Ponder2* [21]. Ponder2 is a toolkit that supports the specification and enforcement of policies in the form of Event-Condition-Action (ECA) rules. When the occurrence of an event is announced to Ponder2, the obligation policy interpreter matches the notification against the registered policy events, hands the event to the relevant policies, which then evaluate their condition(s) and if they succeed, invoke the action(s) specified in the policy.

The Listing 5 shows the XML encoding of an obligation policy that will respond to events of type */event/humanResourcesProcedureEvent*. In the example, Ponder2 checks whether the load on the router R1 is low, and the current time is later than 18:00. In this example, if the conditions are satisfied, the policy invokes an action on the router R1 to reserve 10% of bandwidth in “QoS1” profile.

```

1 <use name="/policy">
2   <add name="AdjustQoSPolicy">
3     <use name="/template/policy">
4       <create type="obligation"
5         event= "/event/humanResourcesProcedureEvent"
6         active="true">
7         <arg name="R1_load"/>
8         <arg name="daytime"/>
9         <condition>
10          <and>
11            <eq>!R1_load;<!-- -->low</eq>
12            <gt>!daytime;<!-- -->18:00</gt>
13          </and>
14        </condition>
15        <action>
16          <use name="/routers/R1">
17            <modify profile="QoS1" value="10%" />
18          </use>
19        </action>
20      </create>
21    </use>
22  </add>
23 </use>

```

Listing 5: An obligation policy

To maintain consistent the state of this policy, the consistency maintenance module receives the datum “*Adjust QoS Policy*” (adj_qos_pol) and its justification list, composed by “*human resources procedure event received*” (hr_proc_evt), “*R1 Low Load Matched*” (R1_load_mat), and “*Daytime Matched*” (dt_mat). The Listing 6 shows the internal representation of this datum and its justifications.

```

1   justification(hr_proc_evt).
2   justification(R1_load_mat).
3   justification(dt_mat).
4   justificationIsPresent(X) :- generated(X).
5   justificationIsPresent(X) :- received(X).
6   datum(adj_qos_pol) :-
7     justificationIsPresent(hr_proc_evt),
8     justificationIsPresent(R1_load_mat),
9     justificationIsPresent(dt_mat).
10  datumIsInternal(adj_qos_pol) :-
11    generated(hr_proc_evt),
12    generated(R1_load_mat),
13    generated(dt_mat).

```

Listing 6: Obligation policy internal representation

We will present a situation as an example of use of our proposal: due to lack of synchronization, some peers disagree about a temporal condition. This situation could lead to to an inconsistent state of node group. In our proposal, after changing the evaluation of the temporal condition, policy processing component informs the belief change to the consistency maintenance module. If the new belief implies a justification change, a message will be sent to peer group informing the change. When receiving this message, the consistency maintenance module of other nodes checks whether the received justification change is consistent with their knowledge base. If it is not, the consistency maintenance module updates the justification and verifies if this update implies in other

changes. The Listing 7 shows the answer from the consistency maintenance module in this situation.

```
1 adj-qos-pol:external (hr_proc_evt:mod R1_load_mat:mod dt_mat:msg)
    Listing 7: A response for the “Adjust QoS Policy” datum
```

The response produced by the consistency maintenance module indicate that the state of the “*Adjust QoS Policy*” (adj-qos-pol) datum is “in” and the option “external” is marked. Besides, the response also shows that the presence of the “*Daytime Matched*” (dt_mat) justification was received as a belief change message (indicate by “:msg” in the Listing7). Thus, the state of the management datum is consistent in node group and the management CDN presents a coherent behavior

The consistency maintenance of state of policies in management CDNs is traditionally performed through centralized entities, such as external repositories [26]. In P2P-based ANM systems, this centralization is normally performed through the use of “super peers” [19] [9]. The utilization of a centralized approach brings concerns in scalability and robustness and imposes difficulties in the integration of different information sources.

It is also possible to use a datum as a justification for another datum (according to Section 3). In our example, we can extend the justification “*R1 Low Load Matched*” to include different “loads” of the router R1, such as memory and CPU load. Thus, we define the datum “*R1 Low Load Matched*” (R1_load_mat) and its justification list, composed by “*CPU Low Load Matched*” (cpu_load_mat) and “*Memory Low Load Matched*” (mem_load_mat). The Listing 8 shows the representation of this datum and its justifications.

```
1 justification(cpu_load_mat).
2 justification(mem_load_mat).
3 justificationIsPresent(X) :- generated(X).
4 justificationIsPresent(X) :- received(X).
5 datum(R1_load_mat) :-
6     justificationIsPresent(cpu_load_mat),
7     justificationIsPresent(mem_load_mat).
8 datumIsInternal(R1_load_mat) :-
9     generated(cpu_load_mat),
10    generated(mem_load_mat).
    Listing 8: A new “R1 Low Load Mat” datum
```

It is also required a little change in the representation of the “*Adjust QoS Policy*” datum. The rule that defines if the option “internal” is marked must be changed as well (*datumIsInternal (adj-qos-pol)*). The Listing 9 shows the new version of “Adjust QoS Policy” datum. The feature presented in the last example, the support for hierarchical justifications, enhances the representation of complex data.

```
1 datum(adj-qos-pol) :-
2     justificationIsPresent(hr_proc_evt),
3     datum(R1_load_mat),
4     justificationIsPresent(dt_mat).
5 datumIsInternal(adj-qos-pol) :-
6     generated(hr_proc_evt),
7     datumIsInternal(R1_load_mat),
8     generated(dt_mat).
    Listing 9: A new “Adjust QoS policy” version
```

Justifications can be used to provide explanations about specific data for users, such as policies [18]. In the network management context, these explanations can be used by human network administrators in order to improve the execution of management tasks and the user comprehension about these tasks. Thus, the utilization of justifications can increase confidence in the results of management tasks (*e.g.*, in policy enforcement) [18].

5.2. Collaborative fault management of links in access networks

The second case study presented is an illustration of the collaborative fault management of interdomain links (*e.g.*, service provider and consumer) through failure notification sent by devices and human knowledge about these notifications (a simpler and shorter version of this case study was described in a previous work [32]). The integration of these information (failure notification in addition to human knowledge) produces a management data, which can assume different states. For example, these data can be used against Service-Level Agreements (SLAs) to support or clarify service level claims.

Among access network technologies in metropolitan networks, Ethernet is one of most interesting and promising choice, thus, we choose this technology to build our case study. In this context, an access network link is an *Ethernet Virtual Connection* (EVC) [28]. Fault management in this link is done through *Alarm Indication Signal* (AIS) messages [28]. These messages are triggered when a link failure occurs. Thus, AIS messages provide asynchronous notification to other elements in the network that there is a fault in the Ethernet network [40]. The efforts to manage layer 2 Ethernet services must consider an overlayed IP infrastructure [38].

In the present case study, nodes have a management service module that collects AIS messages and another module that collects information from human administrators. Besides, there is the consistency maintenance module, responsible to integrate the information from both management service modules and maintain the consistency of the state of management data.

The consistency maintenance module receives the “*Link Fault Detected*” (`linkflt_det`) datum and its list of justifications, composed by “*Service Operator Detection*” (`srv_prv_det`), “*Service Consumer Detection*” (`svr_con_det`), and “*Device Notification Received*” (`dev_not_rcv`). The presence of these justifications is provided by management service modules and kept inside the node group that offers this management service. The Listing 10 shows the representation of this datum and its justifications.

```

1  justification(srv_prv_det).
2  justification(srv_con_det).
3  justification(dev_not_rcv).
4  justificationIsPresent(X) :- generated(X).
5  justificationIsPresent(X) :- received(X).
6  datum(linkflt_det) :-
7      justificationIsPresent(srv_prv_det),
8      justificationIsPresent(srv_con_det),
9      justificationIsPresent(dev_not_rcv).
10 datumIsInternal(linkflt_det) :-
11     generated(srv_prv_det),
12     generated(srv_con_det),
13     generated(dev_not_rcv).
```

Listing 10: “Link Fault Detected” datum

The presence of justifications can be defined by the management service modules or the consistency maintenance module. The “justificationIsPresent” rules represent the possibility to define the presence of justifications through “generated” facts, when the presence is internally generated by the management service modules, and “received” facts, when the presence is received through belief change message from other management nodes (using the consistency maintenance module).

We will present a situation as an example of use of our proposal: due to problem in the reception of an AIS message (represented by the “Device Notification Received” justification), some nodes of the node group disagree about the fault occurrence in a network link (*i.e.*, an EVC). Despite this problem, commands performed by human network administrators in both domains acknowledge the fault and these are received by all the nodes of the node group as the respective justifications (“*Service Operator Detection*” and “*Service Consumer Detection*”). This situation can become the state of the node group inconsistent. However, belief exchange can correct this inconsistency. The Listing 11 shows the answer from the consistency maintenance module in this situation.

```
1 link_flt_det : external (srv_prv_det : mod srv_con_det : mod dev_not_rcv : msg)
```

Listing 11: A response for a “Link Fault Detected” datum

The response produced by consistency maintenance module indicates that the state of the “Link Fault Detected” is “in” and the option “external” is marked. Besides, it can be verified that the presence of the “Device Notification Received” was received through belief change messages (indicated by “:msg” in Listing 11). Thus, the state of this management datum is maintained consistent within the node group and the management CDN presents a coherent behavior.

The *Operations, administration, and maintenance* (OAM) tools of Ethernet enable the use of a signaling intended to differentiate a fault condition and an intentional administrative block in the EVC (*e.g.*, for diagnostic purposes). This signaling is performed by the “*Locked Signal Function*” (LCK) message [38]. In this context, the link fault has to be detected when the LCK message is not perceived. According to the conceptual solution described in Section 3, the definition of a justification that identifies an absence can address the belief about LCK messages.

The justification list of “Link Fault Detected” datum must be modified to introduce a new justification, “*Device Diagnostics Off*” (dev_dia_off). This justification defines that the LCK is absent, therefore the managed EVC is not administrative blocked. The Listing 12 shows a new version of the “Link Fault Detected” datum and its justifications.

```
1 datum(link_flt_det) :-
2   justificationIsPresent(srv_prv_det),
3   justificationIsPresent(srv_con_det),
4   justificationIsPresent(dev_not_rcv),
5   justificationIsPresent(dev_dia_off).
6 datumIsInternal(link_flt_det) :-
7   generated(srv_prv_det),
8   generated(srv_con_det),
9   generated(dev_not_rcv),
10  generated(dev_dia_off).
```

Listing 12: A new “Link Fault Detected” version

Fault management of access links is a management task traditionally performed through stand-alone centralized systems. Usually, these systems are a *network management system* (NMS) (which collects devices notifications); and a *trouble ticket system* (TTS) (which collects information from human network administrators). The traditional procedure brings concerns in scalability and robustness due to the centralization, since each one of these systems is a single point of failure (SPoF) and a bottleneck.

The traditional procedure for fault management of network links also imposes difficulties in the integration of the management information from different sources (NMS and TTS). These systems may operate internally with different data models and languages.

6. Communication strategies

The consistency maintenance module handles the message exchange through CDN communication services. In this process, belief exchange requests are adapted in messages to be spread in management CDN and vice-versa. The management CDN is modeled as an unstructured overlay network, thus there is no relation between the information stored at an management node and its position in the overlay topology. Besides, the set of nodes can be temporally dynamic.

We use the premise that there is support for group organization (*i.e.*, node groups) through management services modules. Thus, nodes that have a specific management service module are organized into a group (without human intervention) and nodes can participate of several groups accordingly to modules that they have. Aggregations are an interesting abstraction to improve the scalability features in distributed systems [44].

The exchange of beliefs about management data is done asynchronously and we do not consider the message exchange to be reliable. It is well known that the utilization of asynchronous unreliable distributed systems imposes challenges to achieve consistency in shared data [11]. Thus, the consistency model used is non-deterministic, in other words, it uses a “weak” notion of consistency. This model is adopted for scalability, robustness, and update dissemination issues. Given a belief “X” that depends on some other belief “Y”, when an update is made to “Y”, it is eventually reflected in “X”. To achieve consistency, all belief changes about a specific datum must be propagated to all nodes that share this datum. Some authors call a similar notion as “eventual consistency” [44].

The methods used for message exchange inside the AMD are modeled using concepts from biology-inspired distributed computing models [2]. Among these models, proliferation-based ones are an interesting choice for communication requirements of our proposal. Several proliferation-based models are described in distributed systems literature. In the present proposal, a belief change possibly undergoes proliferation at nodes visited, where each node calculates the probability of forwarding the belief change using a proliferation controlling function. All nodes in the node group run exactly the same proliferation controlling function and proliferation can be initiated from any node in the node group. We have chosen *replication* as the initial proliferation mechanism in node groups. This mechanism can support several communication strategies. Processes based on replication replication are commonplace in Nature (*e.g.*, epidemic spreading, mammalian immune system) [2].

The communication strategies described in the following Subsections fit with consistency model presented in this Section. Belief computations could occur concurrently and

changes in beliefs are replicated possibly among all nodes within a node group. Different nodes in a node group are not guaranteed to have identical copies of current set of beliefs even if queried at the same time, and not all nodes are guaranteed to perceive each and every update to a current set of beliefs.

In this Section, we present 2 communication strategies developed in the context of the present investigation. Initially, the Unbridled replication is presented in the first Subsection. Afterwards, the controlled replication is present in the second Subsection.

6.1. Unbridled replication

In unbridled replication, management nodes spread messages to replicate every change in justifications among the participating entities (*i.e.*, nodes of a specific node group) [32]. This mechanism is restricted to node group, fulfilling the criterion of robustness for small node groups. Besides, belief changes are replicated as soon as they occur.

The unbridled replication is implemented through the flooding of belief changes inside node groups. For example, flooding techniques have been usually used to implement search operations in unstructured networks. Flooding fulfills the criterion of robustness and also gives fast results in operations inside node groups. However, it produces a huge number of messages which increases the bandwidth used to carry network management traffic and the amount of computational resources necessary to message processing.

The scalability of the unbridled replication is directly related to the definition of node groups, since this operation (message exchange) is restricted to each node group (similarly to *domain*, a concept used in *Astrolabe* [44]). The number of messages within the node group is controlled only through the discard of already received messages. Thus, a node can spread a belief change message, despite having received several messages informing about the same belief change.

6.2. Controlled replication

The *controlled replication* was developed after preliminary investigations regarding the limitations of the unbridled replication for large node groups [34]. Our goal is to use a much lower number of messages in belief changes (than in unbridled replication). Controlled replication forwarding means some belief changes will not be forward due to restriction rules. The idea behind this control is that this way we can minimize redundant network utilization.

The replication is controlled by a *Replication Controlling Function* (RCF). The RCF defines the probability ' P ' of a belief change to be forwarded to the node group. Broadly speaking, this probability could be defined using different parameters of the CDN operation. Initially, we define the RCF as shown in Equation 1:

$$P(\rho, \eta b) = \frac{\rho}{1 + \eta b} \quad (1)$$

where ηb represents the number of received messages of a specific belief change within a "backoff delay" (T) and ρ is the positive proliferation constant. The essence of the function is that proliferation of a belief change should decrease with the reception of multiple copies of the same belief change (message). The equation 1 is evaluated at time T .

When the belief change is produced internally, the node sends belief exchange with $P = 1$ (*i.e.*, the belief change is always sent) to the node group. When the belief change is

received by CDN communication services, the reception of new messages informing the belief change already received during the backoff delay decreases the probability of this belief change to be forwarded to the node group. Note that when $T = 0$ or $\eta b = 0$, the controlled replication has the same behavior of unbridled replication.

7. Simulation Experiments

Scalability and robustness are some of the most important motivations for using decentralization in the infrastructure of different systems [27] [2]. As previously stated, we expect that the introduction of multi-agent truth maintenance features keeps desired properties of a management CDN, maintaining each node as an independent and self-sustainable entity. As many systems have demonstrated, a system that does not share resources can scale almost infinitely simply by adding constitutive elements (*e.g.*, nodes in a management CDN). Besides, maintaining the independence of each node, single points of failure (SPoF) are eliminated.

The evaluation of our proposal can be performed in different ways. To enable a fully controlled environment for the evaluation, we evaluated the system with simulation experiments. A management CDN is modeled as a P2P-based ANM system, thus, the nodes are simulated as peers and node groups as peer groups. In these experiments, we present simulation results that support our scalability and robustness claims.

The simulation experiments were implemented in Java using *PeerSim* [29], an open source event-based simulator of P2P systems. The system version used has the ability to simulate failures in peers and message exchange, and the overlay is built randomly. The experiments use a simple model of transport layer that can emulate some characteristics, such as loss and delay probabilities. In addition, a peer is chosen randomly as the primary source of changes to not affect measurements and message delay is controlled. All peers in peer groups run exactly the same communication strategy. In spite of being event-driven simulations, events are evaluated in cycles of the simulator. We simulate up to 100,000 peers in a peer group.

In this Section, we present results of simulation experiments performed with the communication strategies described in Section 6. First, experiments using unbridled replication in order to provide belief exchange are presented. Then, experiments using controlled replication are also presented.

7.1. Unbridled replication

Unbridled replication is evaluated through 2 simulation experiments. In these experiments, we varied the number of peers of the peer group from 4 to 14 (we do not expect large peer groups using unbridled replication). Each experiment was conducted at least 10 times. In the experiments, the variance observed was low.

In the first experiment, it is measured the number of messages exchanged to spread belief changes in the peer group. This number must be considered as an important cost of the peer group operation, thus, it is important for scalability analysis. Besides, we consider the number of transmitted messages as indicative of network load. In this experiment there were no faults in peers or in message exchange. We show the results in Figure 3.

Our proposal shows acceptable scalability characteristics on number of exchanged messages, since this operation (belief exchange) is restricted to each peer group. The

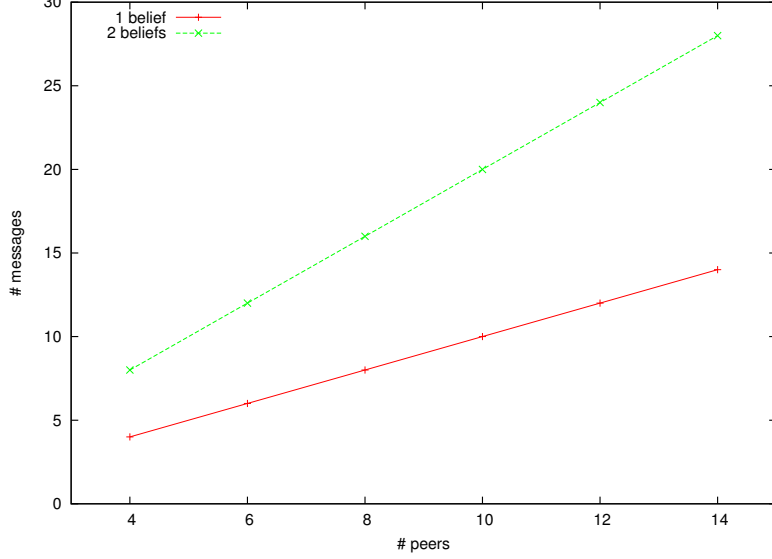


Figure 3: Message exchange due to a belief change - Unbridled Replication

experiment shows that our system behaves like we expected, without stability and convergence problems. Network load grows linearly with the number of participating peers, thus we can infer the behavior trend of peer groups with larger number of participating peers. Of course, an efficient operation of large peer groups needs modifications in the communication strategy, such as using a mechanism to control the replication.

In the second experiment, we determined the influence of message loss on the dissemination of a justification change. In this experiment, we varied the message loss probability with following values: 25%, 50%, and 75% (respectively, 0.25, 0.5, and 0.75 as indicated in Figure 4). Using the case study described in Subsection 5.2, we would probably observe such message loss (specially 75%) due to faulty or overloaded network devices (*e.g.*, ethernet interfaces) and/or network links (*e.g.*, EVCs). Since this case study is aimed at fault management (considering an overlayed IP infrastructure), our proposal must behave acceptably even in bad network conditions. In Figure 4, we show the average percentage of coherent (and correct) peers after message exchange to cease.

The experiment shows the influence of message loss in the replication process. As can be seen from the results in Figure 4, high loss probabilities do lead to less consistency in peer group, but, even with a few participating peers, the percentage of coherent peers is substantial. Besides, more participating peers in peer group decrease the influence of loss probability.

The results show some fault-tolerance features, since the peer group operation is not highly sensitive to peer crashes and message losses. But an increase in number of peers also leads to an important increase in the number of exchanged messages, so the robustness advantages come at some cost. Aggregation of peers is a fundamental abstraction for scalability using unbridled replication as the communication strategy.

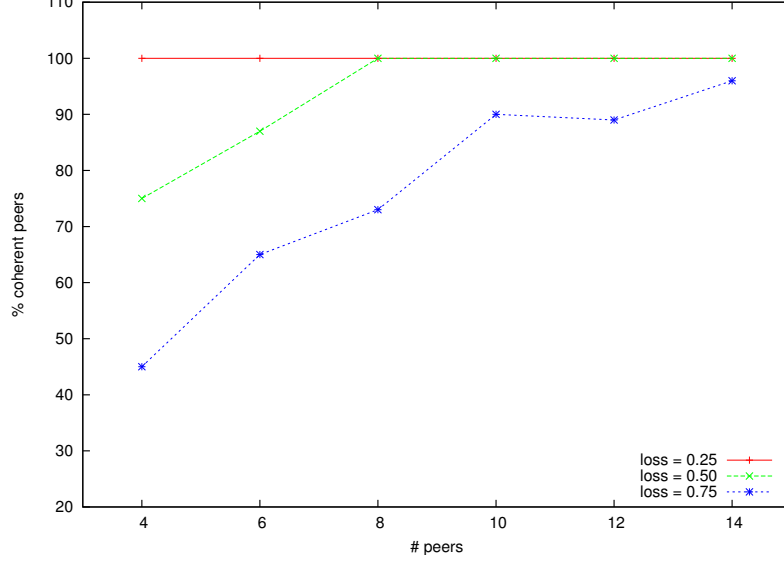


Figure 4: Coherent peers after a belief change - Unbridled Replication

Messages are exchanged only within the peer group, and a high number of peers in a peer group is not expected when this strategy is chosen.

7.2. Controlled replication

Controlled replication is evaluated through 2 simulation experiments. In these experiments, we varied the number of peers of the peer group from 5 to 1000. The parameters used in the Replication Controlling Function (RCF) are informed in each experiment. In the experiments, the variance observed was low.

In the first experiment ($\rho = 1$, $T = 3$), the number of messages exchanged to spread a justification change in the peer group is measured. This number must be considered as an important cost of the peer group operation, thus, it is important for scalability analysis. Besides, we consider the number of transmitted messages as indicative of network load. In this experiment there were no faults in peers or in message exchange. In Figure 5 we report the measured average and 95% confidence intervals for each peer group size.

The experiment shows that our system behaves as we expected, without stability and convergence problems. The main reason for this is that chosen RCF is cost-effective when covering large peer groups. We consider the number of transmitted messages as indicative of network load. Even though we do not expect this situation in P2P-based ANM systems (and in other management CDNs), it is important to perform this experiment to infer the behavior trend of peer groups with larger number of participating peers.

In the second experiment, we determined the influence of message loss on the dissemination of a justification change. In this experiment, we varied the message loss probability with following values: 50%, and 75%. During the experiments, we observed a significantly correlation between message loss and positive proliferation constant (ρ) on

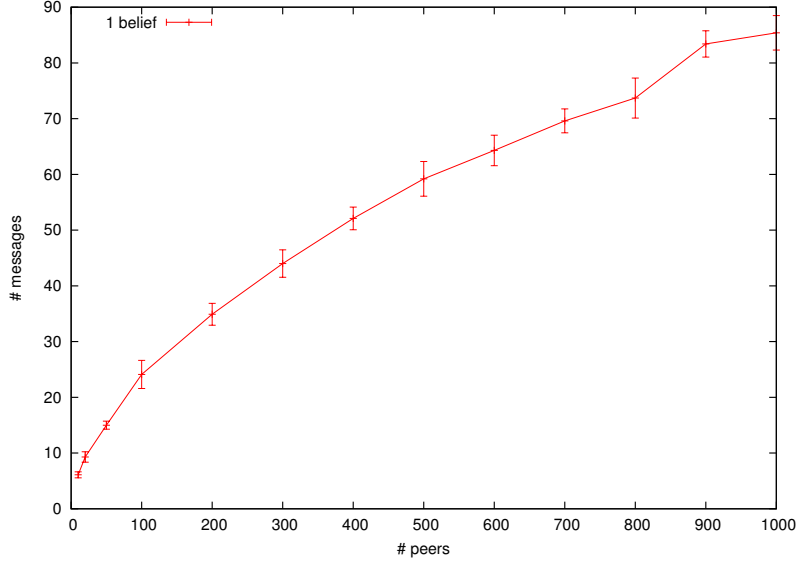


Figure 5: Message exchange due to a belief change - Controlled replication

results, thus, we also varied ρ in this experiment with following values: 0.5 and 0.25. In Figure 6, we show the average percentage of coherent (and correct) peers after message exchange to cease.

The experiment shows the influence of message loss and positive proliferation constant (ρ) in the controlled replication scheme. As can be seen from the results in Figure 6, high loss probabilities do lead to less consistency in peer group, but, even with a few participating peers, the percentage of coherent peers is substantial. Besides, more participating peers in peer group and higher values of ρ decrease the influence of loss probability.

8. Related work

In recent years, several research efforts on services for consistency of shared information have been carried out distributed systems community. These services can be used as a basic building block for distributed applications. A management CDN can be viewed as a distributed application and it can be improved with characteristics found in these services. In this section, we cover some of the most prominent investigations.

ZooKeeper [16] is a coordination service for distributed applications. It exposes a simple API that distributed applications can be built upon to implement higher level services for synchronization, data diffusion, and publish-subscribe schemes. *ZooKeeper* use distributed server databases for read operations, however, write operations use a “leader” server (*i.e.*, centralized database) to assure the consistency of the database.

Astrolabe [44] is a distributed information management service. It works locating and collecting the status of a set of servers and reporting summaries of this information.

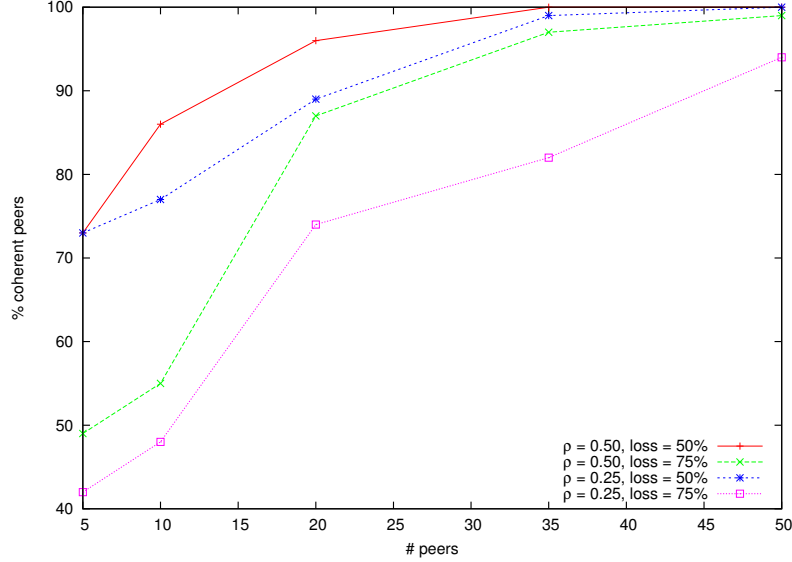


Figure 6: Coherent peers after a belief change - Controlled replication

Astrolabe is implemented using a P2P overlay, where every peer runs an *Astrolabe* agent (*i.e.*, in a MAS fashion). However, *Astrolabe* was developed primarily using simple data models. Besides, its operation is aimed at read-oriented applications.

Scalable Distributed Information Management System (SDIMS) [46] is a service to aggregate information about large-scale network systems. The service is built using ideas from *Astrolabe* [44] and Distributed Hash Tables (DHT). However, as in most DHT approaches, consistency and replication issues are a known challenge.

The presented efforts show interesting characteristics for consistency of shared information in distributed systems. However, these efforts have vulnerabilities which make them not appropriate for management CDNs, such as centralization [16], simple data models [44], and replication issues [46].

9. Final remarks

The support of new demands faced by traditional network management is a key research issue in the network management area. The utilization of Complex Dynamic Networks (CDNs) can be a feasible approach for these demands, specially when it is used together with automation features. However, the consistency of state of management imposes challenges for management CDNs.

In this paper we use multi-agent truth maintenance features and dynamic process as communication strategies to improve the consistency of state of management data in management CDNs. Our proposal aims at the integration of data used by the entities that form these systems (*i.e.*, nodes), through the utilization of belief about management data. We have also presented evaluations of this proposal through simulation experiments. In addition, we have described case studies to show the possibilities of our proposal.

Although the proposal shows good results in evaluations performed until the present moment, it is necessary to evaluate more complicated cases, in number of nodes and node groups, and in the participation of an nodes in different node groups. We are also looking at additional settings that could lead to important effects, such as network partitions. Thus, we are currently pursuing new experiments with *PeerSim*.

References

- [1] R. Albert, H. Jeong, and A. Barabási. Attack and error tolerance of complex networks. *Nature*, 406(6794):378–382, 2000.
- [2] O. Babaoglu, G. Canright, A. Deutsch, G. Di Caro, F. Ducatelle, L. Gambardella, N. Ganguly, M. Jelasity, R. Montemanni, A. Montresor, et al. Design patterns from biology for distributed computing. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 1(1):26–66, 2006.
- [3] N. Badr, A. Taleb-Bendiab, and D. Reilly. Policy-based autonomic control service. In *Policies for Distributed Systems and Networks, 2004. POLICY 2004. IEEE Workshop on*, pages 99–102, 2004.
- [4] K. Clark and F. McCabe. Ontology schema for an agent belief store. *International Journal of Human-Computer Studies*, 65(7):640–658, 2007.
- [5] Y. Dimopoulos and P. Moraitis. Multi-agent coordination and cooperation through classical planning. *Intelligent Agent Technology, 2006. IAT '06. IEEE/WIC/ACM International Conference on*, pages 398–402, Dec. 2006.
- [6] S. Dobson, S. Denazis, A. Fernández, D. Gaïti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, and F. Zambonelli. A survey of autonomic communications. *ACM Trans. Auton. Adapt. Syst.*, 1(2):223–259, 2006.
- [7] J. Doyle. Truth maintenance systems for problem solving. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1978.
- [8] J. Doyle. A truth maintenance system. *Computation & intelligence: collected readings*, pages 529–554, 1979.
- [9] L. Fallon, D. Parker, M. Zach, M. Leitner, and S. Collins. Self-forming Network Management Topologies in the Madeira Management System. *Lecture Notes in Computer Science*, 4543:61, 2007.
- [10] M. Fischer, N. Lynch, and M. Merritt. Easy impossibility proofs for distributed consensus problems. *Distributed Computing*, 1(1):26–39, 1986.
- [11] M. Fischer, N. Lynch, and M. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.
- [12] L. Gong. Jxta: a network programming environment. *IEEE Internet Computing*, 5(3):88–95, May/Jun 2001.
- [13] L. Z. Granville, D. M. da Rosa, A. Panisson, C. Melchior, M. J. B. Almeida, and L. M. R. Tarouco. Managing computer networks using peer-to-peer technologies. *IEEE Communications Magazine*, 43(10):62–68, 2005.
- [14] B. Horling and V. Lesser. A survey of multi-agent organizational paradigms. *The Knowledge Engineering Review*, 19(04):281–316, 2005.
- [15] M. N. Huhns and D. M. Bridgeland. Multiagent truth maintenance. *Systems, Man and Cybernetics, IEEE Transactions on*, 21(6):1437–1445, 1991.
- [16] Hunt, P. *ZooKeeper: A Distributed Coordination Service for Distributed Applications*. <http://wiki.apache.org/hadoop/ZooKeeper>, 2010.
- [17] B. Jennings, S. Van Der Meer, S. Balasubramaniam, D. Botvich, M. O. Foghlu, W. Donnelly, and J. Strassner. Towards autonomic management of communications networks. *IEEE Communications Magazine*, 45(10):112–121, 2007.
- [18] L. Kagal, C. Hanson, and D. Weitzner. Using Dependency Tracking to Provide Explanations for Policy Management. In *Policies for Distributed Systems and Networks, 2008. POLICY 2008. IEEE Workshop on*, pages 54–61, 2008.
- [19] C. Kamiński, J. Fidalgo, R. Dantas, D. Sadok, and B. Ohlman. Design and implementation of a policy-based management framework for ambient networks: Choices and lessons learned. *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, pages 775–778, April 2008.
- [20] C. Kamiński, J. Fidalgo, D. Sadok, J. Lima, L. Pereira, and B. Ohlman. PBMAN: A Policy-based Management Framework for Ambient Networks. In *Policies for Distributed Systems and Networks, 2006. POLICY 2006. IEEE Workshop on*, pages 79–83, 2006.

- [21] Kevin Twidle. *Ponder2 Documentation*. URL <http://www.ponder2.net/>, 2010.
- [22] G. Lawton. Machine-to-machine technology gears up for growth. *Computer*, 37(9):12–15, 2004.
- [23] F. Lorenzi. A multiagent knowledge-based recommender approach with truth maintenance. *Proceedings of the 2007 ACM conference on Recommender systems*, pages 195–198, 2007.
- [24] E. Lupu, N. Dulay, M. Sloman, J. Sventek, S. Heeps, S. Strowes, K. Twidle, L. Keoh, and A. SchaefferFilho. Amuse: autonomic management of ubiquitous systems for e-health. *J. Concurrency and Computation: Practice and Experience, John Wiley (to appear)*, 2007.
- [25] C. C. Marquezan, C. R. P. dos Santos, J. C. Nobre, M. J. B. Almeida, L. M. R. Tarouco, and L. Z. Granville. Self-managed services over a p2p-based network management overlay. In *Proceedings. 2nd Latin American Autonomic Computing Symposium (LAACS 2007)*, 2007.
- [26] C. C. Marquezan, A. Panisson, L. Z. Granville, G. Nunzi, and M. Brunner. Maintenance of monitoring systems throughout self-healing mechanisms. In *19th IFIP/IEEE international workshop on Distributed Systems: Operations and Management, Proceedings*, page 188. Springer, 2008.
- [27] J. A. Mccann and M. C. Huebscher. Evaluation Issues in Autonomic Computing. In *Grid and Cooperative Computing – GCC 2004*, volume 3252 of *Lecture Notes in Computer Science*, pages 597–608. Heidelberg, Springer-Berlin, 2004.
- [28] M. McFarland, S. Salam, and R. Checker. Ethernet oam: key enabler for carrier class metro ethernet services. *IEEE Communications Magazine*, 43(11):152–157, 2005.
- [29] A. Montresor and M. Jelasity. PeerSim: A scalable P2P simulator. In *Proc. of the 9th Int. Conference on Peer-to-Peer (P2P’09)*, 2009.
- [30] R. Moore. *Creating a DCOP Interface*. URL <http://developer.kde.org/documentation/>, 2010.
- [31] M. E. J. Newman. The Structure and Function of Complex Networks. *SIAM Review*, 45(2):167–256, 2003.
- [32] J. Nobre and L. Granville. Consistency of States of Management Data in P2P-Based Autonomic Network Management. In *20th Ifip/Ieee International Workshop on Distributed Systems: Operations and Management, Dsom 2009, Venice, Italy, October 27-28, 2009, Proceedings*, pages 99–110. Springer-Verlag New York Inc, 2009.
- [33] J. C. Nobre and L. Z. Granville. Towards consistency of policy states in decentralized autonomic network management. In *Policies for Distributed Systems and Networks, IEEE International Workshop on*, volume 0, pages 170–173, Los Alamitos, CA, USA, 2009. IEEE Computer Society.
- [34] J. C. Nobre and L. Z. Granville. Consistency of Policy States in Decentralized Autonomic Network Management. In *Proceedings of the 12th IEEE/IFIP Network Operations and Management Symposium (NOMS 2010), Osaka, Japan. April 2010.*, 2010.
- [35] A. Panisson, D. M. da Rosa, C. Melchior, L. Z. Granville, Maria, and Liane. Designing the Architecture of P2P-Based Network Management Systems. In *ISCC ’06: Proceedings of the 11th IEEE Symposium on Computers and Communications*, pages 69–75. IEEE Computer Society, 2006.
- [36] H. Pennington. *D-bus specification*. URL <http://dbus.freedesktop.org/doc/dbus-specification.html>, 2010.
- [37] A. Pras, J. Schoenwaelder, M. Burgess, O. Festor, G. M. Perez, R. Stadler, and B. Stiller. Key research challenges in network management. *IEEE communications magazine*, 45(10):104–110, October 2007.
- [38] J. Ryoo, J. Song, J. Park, and B. Joo. Oam and its performance monitoring mechanisms for carrier ethernet transport networks. *Communications Magazine, IEEE*, 46(3):97–103, 2008.
- [39] N. Samaan and A. Karmouch. Towards autonomic network management: An analysis of current and future research directions. *IEEE Communications Surveys and Tutorials*, 11(3), 2009.
- [40] R. Sanchez, L. Raptis, and K. Vaxevanakis. Ethernet as a carrier grade technology: developments and innovations. *IEEE Communications Magazine*, 46(9):88–94, September 2008.
- [41] R. Scowen. Technical Report ISO/IEC DIS 13211-1: 1995 (E). Technical report, International Organisation for Standardisation, Geneva, Switzerland, 1995.
- [42] S. Strogatz. Exploring complex networks. *Nature*, 410(6825):268–276, 2001.
- [43] K. P. Sycara. Multiagent systems. *AI Magazine*, 19(2):79–92, 1998.
- [44] R. Van Renesse, K. Birman, and W. Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM Transactions on Computer Systems (TOCS)*, 21(2):164–206, 2003.
- [45] I. Warfield, C. Hogg, S. Lee-Urban, and H. Munoz-Avila. Adaptation of hierarchical task network plans. *Proceedings of the Twentieth International FLAIRS Conference (FLAIRS-07)*, 2007.
- [46] P. Yalagandula and M. Dahlin. A scalable distributed information management system. *ACM SIGCOMM Computer Communication Review*, 34(4):379–390, 2004.